FACULTY WORKING PAPER 93-0110

Optimal Sequential File Search: A Reduced-State Dynamic Programming Approach

Charles Blair
Department of Business Administration
University of Illinois

George E. Monahan
Department of Business Administration
University of Illinois

Bureau of Economic and Business Research
College of Commerce and Business Administration
University of Illinois at Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 93-0110

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

January 1993

Optimal Sequential File Search: A Reduced-State Dynamic Programming Approach

Charles Blair

George E. Monahan

Department of Business Administration

Optimal Sequential File Search: A Reduced-State Dynamic Programming Approach

Charles Blair and George E. Monahan [1]

Department of Business Administration
University of Illinois at Urbana-Champaign
Champaign, Illinois 61820

January, 1993

Abstract

This paper continues the study of a file search problem in Monahan [4]. The objective is to determine whether or not a record is in a sequential file whose contents are initially unknown. Costly information regarding the contents of positions within the file can be purchased. The problem of determining an optimal search and disposition strategy is formulated as a Markov decision process whose state space is polynomial in the number of possible records in the file. As a result, the computational effort required to determine transition probabilities and expected payoffs is also polynomial.

Key Words: Finite State Markov Decision Processes (116), Search (751)

Please address all correspondence to:
Professor George E. Monahan
Department of Business Administration
1206 South Sixth Street
Champaign, IL 61820

[1]Part of this research was conducted while the second author was on sabbatical leave at the London School of Economics and Political Science. He would like to thank the Operational Research group in the Department of Statistical and Mathematical Sciences at LSE for its support.

1 Introduction

This paper continues the study of a file searching problem analyzed in Monahan [4]. The file consists of a subset of records that have been randomly selected from a given population of records. The probability of selecting each record in the population is known. Records are stored sequentially on the basis of some key value. We are concerned with whether a given record is or is not in the file. We can declare the status of the record immediately or we can gather costly information about the contents of the file by examining as many positions within the file as we wish and then making the declaration. A reward is earned if this declaration is correct. The objective is to determine a strategy for acquiring information and making a declaration that maximizes the expected terminal payoff less the total cost of acquiring information.

Monahan [4] models this problem as a Markov decision process (MDP) whose state space is the set of probability distributions over the set of all possible file contents. In this paper, we propose another MDP whose state space is considerably smaller, which leads to a polynomial-time algorithm for solving the problem.

The paper is organized as follows. An explicit statement of the problem is given in the next section. Section 3 discusses the relationship between the model developed here and other models of search. The reduced-state MDP is given in Section 4. Section 5 derives explicit expressions for the state transition probabilities. A numerical example is given in Section 6. Concluding remarks are in Section 7.

2 The Problem

There are n possible records, each uniquely labelled by an integer between 1 and n. The file includes record i with probability p_i. The selection of any record is stochastically independent of the selection of other records. The members of the file are sorted with the first position in the file being the smallest included record. For example, the probability that the first position contains record 3 is $(1-p_1)(1-p_2)p_3$. The total number of included records is a random variable (which is the sum of n Bernoulli random variables with parameters $p_1, \ldots, p_n$, respectively), whose value is initially unknown.

A query asks for the contents of the I-th position in the file, which is the I-th smallest included record. It costs \$$c$ to make each query. (Note: Capital letters will be used for positions in the file. Small letters refer to the value of keys of records in the file. For example, we refer to record h in position I.) We are either told which record is in position I, or that fewer than I records have been included (i.e., position I is empty). A *negative query* is a query that indicates that position I is empty.

We wish to determine whether record h has been included in the file. At any stage of the decision process, we make a query or terminate the search by either:

I. Claiming record h is in the file but not specifying a specific position or

II. Claiming record h has not been included in the file and that there are exactly L records in the file smaller than h.

If the disposition claim is correct, a reward of \$$r$ is earned.

3 Related Literature

Monahan [4] discusses the relationship between the file search model studied here and other search models, including file search models, that are in the computer science, economics, and operation research literatures. For completeness, the highlights of that discussion are included here.

Wiederhold [8] describes several techniques for searching a sequential computer file but makes no assumptions regarding the likelihood of certain records being in the file. The model studied here can be viewed as Bayesian version of the binary search and probing schemes he discusses.

There is an extensive literature dealing with methods for searching sequential computer files. See Knuth [2, 3] for a discussion of “classical” file search procedures. The classical problem differs from the problem studied here. In the classical problem, the records in a sequential file are known. A record is randomly drawn with a known probability from a given population. Its key cannot be observed. The objective is to determine if this record is in the file. The only way to make this determination is to compare the record to existing records in the file. If, after a comparison, the keys match, the record is in the file. If they do not, additional comparisons are made. An optimal search procedure prescribes the sequence of comparisons that optimizes some objective. See Moore, et al. [5] and Moore and Whinston [6, 7] for a decision-theoretic discussion of this form of the problem.

The assumptions regarding what is known and not known are reversed in the problem studied here. We have one record whose key is *known* and wish to determine if it is in a file consisting of records whose keys are initially *unknown*.

The search problem studied here applies to settings other than sequential computer files. Indeed it applies to any problem that is characterized by entities randomly drawn from a known population so that certain of them may or may not be present. Those that are present, however, always appear in a known order. Possible examples include the videotape of a sporting event or the geologic structure of a potential drilling site for an oil well.

4 A Stochastic Dynamic Program

We wish to determine whether record h is in the file. (Monahan [4] refers to this as Problem h.) We have examined some of the positions in the file and know that records i and k are in positions I and $I + M + 1$, respectively, with $i < h < k$, but we have no information regarding the M records between i and k. That is, record i (k) is the greatest (least) record known to be in the file that is smaller (greater) than h. The dynamic program developed in this section exploits the fact that *all subsequent decisions need only take into account* i, k, *and* M.

The decision process consists of two phases. In the *search phase*, positions in the file are examined at a cost of \$$c$ per position. At any stage of the decision process, a *search strategy* specifies the next position to be examined based upon all of the information available at the beginning of that stage. After examining a position, the state of the decision process is updated to reflect the information that has just been obtained. The process then proceeds optimally.

The second phase of the decision process is the *disposition phase* that entails the choice of one of several stopping actions. Based upon the state of the process, record h is declared either to be in the file or not in the file. The *disposition strategy* specifies which of the stopping actions given in I and II above are to be taken. A correct disposition decision earns

\$$r$. The objective is to find a search and disposition strategy that maximizes the expected terminal reward less the total cost of examining the file. Such a strategy is called optimal.

We formulate the problem of searching a sequential file for record h as a stochastic dynamic program (Markov decision process). For notational convenience, the dependence of the elements of the model on h is suppressed.

The state space of the decision process is

$$\mathcal{S} = \{(i,k,M) \,|\, i = 0,\ldots,h-1,\ k = h+1,\ldots,n+1, \text{ and } M = 0,\ldots,k-i-1\}.$$

We add to the original n possible records, the fictitious record 0, which is smaller than all other records. We specify $p_0 = 1$, so that record 0 is known to occupy position 0 before any queries are made. Thus a state with $i = 0$ indicates we have not yet observed any (non-fictitious) record smaller than h in the file.

By convention, $k = n+1$ labels the condition where we have not yet observed any records larger than h. When $k = n+1$, M denotes the *maximum* number of records that could possibly be in the file. Thus the initial state of the decision process is $(0, n+1, n)$. If our first query reveals that there is no record in position J, we will then be in state $(0, n+1, J-1)$.

At times, it will be convenient to say that record $n+1$ (considered larger than record h) was observed in a given file position, rather than saying the query on that position was negative.

At any stage of the decision process, one of several actions can be taken. Suppose, for example, that record i is in position I and that the state is $(i,k,M) \in \mathcal{S}$. One possible action is to query position $I+L$, for $L = 1,\ldots,M$, at a cost of \$$c$. Suppose that record j is observed in this position. The new state of the decision process depends both upon j and the current state. Potential new states include $(j,k,M-L)$, which occurs if $j < h$, and $(i,j,L-1)$ if $j > h$. (Note that this is consistent with our conventions in the case $j = k = n+1$.) The probability that the query in position $I+L$ shows record k is denoted by $p(j|i,k,M,L)$. This is a transition probability between the state (i,k,M) and either $(j,k,M-L)$ or $(i,j,L-1)$ as described above. In the next section, these probabilities are expressed in terms of $p_1,\ldots,p_n$—the underlying probabilities that records $1,\ldots,n$ are in the file. After making a query, the decision process continues for at least one more stage.

Other actions, all of them associated with stopping the decision process, are possible.

When a disposition action is taken, the expected terminal reward, whose value depends upon the current state, is received and the decision process terminates. Option II requires that we specify the "gap" in which record h falls. Suppose that the state is $(i, k, M) \in \mathcal{S}$ and, without loss of generality, that record i is in position I. We say record h is in Gap L if the record in position $I + L$ is less than record h and the record in position $I + L + 1$ is greater than h, for $L = 0, \ldots, M$. In the specific case in which all queries have received negative responses ($i = 0$ and $k = n + 1$), record h is said to be in Gap 0 if the file is empty or if all records in the file are greater than h. This is consistent with Monahan [4].

Let $V(i, k, M)$ denote the maximum expected payoff that can be received if the state of the decision process is $(i, k, M) \in \mathcal{S}$ and the process continues from the next stage onwards optimally. Then $V(\cdot, \cdot, \cdot)$ satisfies the following dynamic programming recursion. For $(i, k, M) \in \mathcal{S}$,

$$V(i,k,M) = \max \begin{cases} J(i,k,M,L) & \text{expected reward if the } L\text{-th record following record } i \text{ is examined and the process proceeds optimally, } L = 0, \ldots, M \\ S(i,k,M) & \text{expected (terminal) reward if the search process is terminated and record } h \text{ is declared to be in the file} \\ G(i,k,M,L) & \text{expected (terminal) reward if the search process is terminated and record } h \text{ is declared to be in gap } L, L = 0, \ldots, M \end{cases} \tag{1}$$

where

$$\begin{aligned} J(i,k,M,L) &= -c + rp(h|i,k,M,L) + \sum_{j=i+1}^{h-1} p(j|i,k,M,L)\, V(j,k,M-L) \\ &\quad + \sum_{j=h+1}^{k} p(j|i,k,M,L)\, V(i,j,L-1), \end{aligned} \tag{2}$$

for $L = 1, \ldots, M$. The first term in J is the cost of the query. The second, third, and fourth terms are the optimal expected benefits that can be obtained if a query reveals that the record in "location" L is equal to, smaller, or larger, than h, respectively.

Under Option I,

$$S(i,k,M) = r\sum_{L=1}^{M} p(h|i,k,M,L).$$

Under Option II,

$$G(i,k,M,L) = r\sum_{j=i}^{h-1}\left[p(j|i,k,M,L)\cdot\sum_{t=h+1}^{k} p(t|j,k,M-L,1)\right],$$

for $L = 0,\ldots,M$.

In the next section, we derive an explicit representation for the transition probabilities $p(j|i,k,M,L)$ in terms of the underlying probabilities $p_1,\ldots,p_n$. We then illustrate this procedure with a small numerical example used in [4].

5 The State Transition Probabilities

Recall that p_i is the probability that record i is included in the file. Let $q_i = 1 - p_i$. The probabilities presented in this section are computed using Bayes' rule. It is convenient to consider two mutually exclusive cases representing information about the file. In all cases, we assume $(i,k,M) \in \mathcal{S}$ is the state of decision process.

5.1 Case 1: $k < n+1$

In this case, we have observed at least one record greater than h. We also know that there are exactly M records between records i and k.

Let $s(i,k,M)$ denote the probability that records i and k are in the file, as well as exactly M of records $i+1$ through $k-1$. These probabilities are computed recursively as follows:

$$s(i,k,0) = p_i p_k \prod_{j=i+1}^{k-1} q_j \quad \text{and} \quad s(i,k,M+1) = p_i \sum_{j=i+1}^{k-1}\Big(\prod_{m=i+1}^{j-1} q_m\Big)s(j,k,M),$$

with the convention that the product over an empty index set is 1. The (posterior) probability that record j is the L-th largest record in the file following record i is

$$p(j|i,k,M,L) = \frac{s(i,j,L-1)s(j,k,M-L)p_j^{-1}}{s(i,k,M)}. \tag{3}$$

5.2 Case 2: $k = n + 1$

In this case record $i < h$ is the largest record located by query and M is the maximum number of records that are greater than i. (Since there are n possible records, $M \leq n - i$, with inequality holding if a query has received a negative answer.)

For $g < n+1$, let $u(g, N)$ denote the (prior) probability that the file includes record g and at most N records greater than g. As before, these probabilities can be defined recursively as follows:

$$\begin{aligned} u(g,0) &= p_g \prod_{j=g+1}^{n} q_j \quad \text{and} \\ u(g,N+1) &= p_g \sum_{j=g+1}^{n} \Big(\prod_{m=g+1}^{j-1} q_m \Big) u(j,N) + p_g \prod_{j=g+1}^{n} q_j . \end{aligned}$$

For $j < n+1$, the (posterior) probability that record j is the L-th smallest record included in the file after record i is

$$p(j|i, n+1, M, L) = \frac{s(i,j,L-1)u(j,M-L)p_j^{-1}}{u(i,M)}. \tag{4}$$

The chance that the query in position L is negative is:

$$p(n+1\,|\,i, n+1, M, L) = 1 - \sum_{j=i+1}^{n} p(j\,|\,i, n+1, M, L).$$

6 A Numerical Example

The following numerical example is given in Monahan [4]: $n = 3$, $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$, $r = 1$, and $c = 0.1$. The problem is to determine the status of record in the population whose key is 2, so that $h = 2$.

There are a number of standard procedures for computing the solution to the reduced-state MDP given in (1); see, for example, Heyman and Sobel [1, Section 4–5]. For expositional purposes, we use one of the linear programming approaches. Constraints come in "bundles", one bundle per state, and one constraint for every action that is feasible in that state. At least one slack variable will be zero for constraints in a given bundle. The action associated with this constraint is an optimal action to take in the state associated with the bundle in which that constraint appears. The complete formulation is given in Appendix 1.

6.1 The Optimal Search and Disposition Strategy

We know that following variables are equal to one since they correspond to states representing complete certainty regarding the contents of the file:

$$V(0,4,0) = V(0,3,2) = V(1,3,0) = V(1,4,0) = V(1,3,1) = V(0,3,0) = 1.$$

We substitute these values into the linear program, as well as the values for the posterior probabilities that are determined using (3)-(4) and obtain the explicit form of the linear program given in Appendix 2. In this formulation, $V(i,k,M)$ is written as $Vikm$, for $(i,k,M) \in \mathcal{S}$.

The optimal search and disposition strategy is determined by the value of the slack variables in the optimal solution to the linear program. For convenience, the rows whose slack variables are zero are labelled with their corresponding state-action pair.

The optimal expected payoffs associated with states in $\mathcal{S}$ for which there is uncertainty regarding the status of record 2 are:

$$V031 = V041 = V141 = V142 = 0.9,\ V042 = 0.8299, \text{ and } V043 = 0.89.$$

The following table lists each state-action pair for which the associated slack variable is zero in the optimal solution.

		Optimal
Row	State	Action
2)	(0,3,1)	$L = 1$
5)	(0,4,1)	$L = 1$
9)	(0,4,2)	$L = 1$
14)	(1,4,1)	$L = 1$
17)	(0,4,3)	$L = 1$
23)	(1,4,2)	$L = 1$

Table 1: An Optimal Search and Disposition Strategy

The expected payoff, given we know nothing about the contents of the file and follow an optimal search and disposition strategy, is 0.89, since the initial state of the decision process is $(0,4,3)$. Using Table 1, the optimal search and disposition strategy is the following:

- Examine the first position of the file. The initial state is $(0, 4, 3)$ and, from Row 17 of Table 1, the optimal action to take is $L = 1$ (which refers to the position 1 in the file).
- If a "1" is observed in position 1, examine position 2. Record 1 in position 1 corresponds to state (1,4,2). From Row 23 in Table 1, the optimal action is $L = 1$, indicating that it is optimal to examine the position immediately following record 1, which is position 2. An examination of the second position will determine the status of record 2 with certainty.
- If anything other than "1" is observed in the first position, the status of record 2 is known with certainty.

This is, of course, the same strategy that was determined by a different procedure in [4].

7 Concluding Remarks

The problem of searching for a record in a sequential file was formulated as a finite-state and finite-action Markov decision process. We refer to this formulation as a "reduced-state" model since the number of states in the state space is polynomial ($O(n^3)$) in the size of the underlying population. It is easy to see that the transition probability calculations in Section 5 and the expected payoff calculations in Section 4 can be carried out in polynomial time, so we have a polynomial-time algorithm for finding the optimal strategy.

More general versions of the model are possible. We could give the player the option of identifying the exact location of record h for a larger reward, or claiming the record is not in the file without identifying the gap for a smaller reward. The state space and transition probabilities would be the same as for the original problem, but the expected payoff calculations would have to include the additional options. It would also be possible to make the cost of queries and rewards time-dependent by adding a time dimension to the state space. An interesting, but more difficult, generalization would permit multiple copies of records to be in the file.

Appendix 1: A Linear Program

The state and action associated with each constraint is noted in the following linear program. For notational convenience, s is used to represent the current state in the expressions for the transition probabilities in each block of constraints.

$$\text{Min} \sum_{(i,k,M)\in\mathcal{S}} V(i,k,M)$$

Subject to:

State	Action	Inequality
s = (1,4,1)	$L = 1$	$V(1,4,1) \geq -c + p(2\vert\mathsf{s},1) + p(3\vert\mathsf{s},1)\,V(1,3,0) + (1 - p(2\vert\mathsf{s},1) - p(3\vert\mathsf{s},1))\,V(1,4,0)$
	In	$V(1,4,1) \geq p(2\vert\mathsf{s},1)$
	Gap 1	$V(1,4,1) \geq 1 - p(2\vert\mathsf{s},1)$
s = (0,4,1)	$L = 1$	$V(0,4,1) \geq -c + p(2\vert\mathsf{s},1) + p(1\vert\mathsf{s},1)\,V(1,4,0) + (1 - p(1\vert\mathsf{s},1) - +p(2\vert\mathsf{s},1) - p(3\vert\mathsf{s},1))\,V(0,4,0) + p(3\vert\mathsf{s},1)\,V(0,3,0)$
	In	$V(0,4,1) \geq p(2\vert\mathsf{s},1)$
	Gap 0	$V(0,4,1) \geq 1 - p(1\vert\mathsf{s},1) - p(2\vert\mathsf{s},1)$
	Gap 1	$V(0,4,1) \geq p(1\vert\mathsf{s},1)$
s = (0,4,2)	$L = 1$	$V(0,4,2) \geq -c + p(2\vert\mathsf{s},1) + p(1\vert\mathsf{s},1)\,V(1,4,1) + (1 - p(1\vert\mathsf{s},1) - p(2\vert\mathsf{s},1) - p(3\vert\mathsf{s},1))\,V(0,4,0) + p(3\vert\mathsf{s},1)\,V(0,3,0)$
	$L = 2$	$V(0,4,2) \geq -c + p(2\vert\mathsf{s},2) + p(3\vert\mathsf{s},2)\,V(0,3,1) + (1 - p(2\vert\mathsf{s},2) - p(3\vert\mathsf{s},2))\,V(0,4,1)$
	In	$V(0,4,2) \geq p(2\vert\mathsf{s},1) + p(2\vert\mathsf{s},2)$
	Gap 0	$V(0,4,2) \geq 1 - p(1\vert\mathsf{s},1) - p(2\vert\mathsf{s},1)$
	Gap 1	$V(0,4,2) \geq p(1\vert\mathsf{s},1)(1 - p(2\vert\mathsf{s},2))$
s = (0,4,3)	$L = 1$	$V(0,4,3) \geq -c + p(2\vert\mathsf{s},1) + p(1\vert\mathsf{s},1)\,V(1,4,2) + (1 - p(1\vert\mathsf{s},1) - p(2\vert\mathsf{s},1) - p(3\vert\mathsf{s},1))\,V(0,4,0) + p(3\vert\mathsf{s},1)\,V(0,3,2)$
	$L = 2$	$V(0,4,3) \geq -c + p(2\vert\mathsf{s},2) + p(3\vert\mathsf{s},2)\,V(0,3,1) + (1 - p(2\vert\mathsf{s},2) - p(3\vert\mathsf{s},2))\,V(0,4,1)$
	$L = 3$	$V(0,4,3) \geq -c + p(3\vert\mathsf{s},3)\,V(0,3,2) + (1 - p(3\vert\mathsf{s},3))\,V(0,4,2)$
	In	$V(0,4,3) \geq p(2\vert\mathsf{s},1) + p(2\vert\mathsf{s},2)$
	Gap 0	$V(0,4,3) \geq 1 - p(1\vert\mathsf{s},1) - p(2\vert\mathsf{s},1)$
	Gap 1	$V(0,4,3) \geq p(1\vert\mathsf{s},1)(1 - p(2\vert\mathsf{s},2))$

State	Action	Inequality	
$s = (0,3,1)$	$L = 1$	$V(0,3,1) \geq$	$-c + p(2\|s,1) + p(1\|s,1)V(1,3,0)$
	In	$V(0,3,1) \geq$	$p(2\|s,1) + p(2\|s,2)$
	Gap 1	$V(0,3,1) \geq$	$p(1\|s,1)$
$s = (1,4,2)$	$L = 1$	$V(1,4,2) \geq$	$-c + p(2\|s,1) + p(3\|s,1)V(1,3,0)$ $+ (1 - p(2\|s,1) - p(3\|s,1))V(1,4,0)$
	$L = 2$	$V(1,4,2) \geq$	$-c + p(3\|s,2)V(1,3,1) + (1 - p(3\|s,2))V(1,4,1)$
	In	$V(1,4,2) \geq$	$p(2\|s,1)$
	Gap 1	$V(1,4,2) \geq$	$1 - p(2\|s,1)$

Appendix 2. The Explicit Linear Program.

min $V031 + V041 + V042 + V141 + V043 + V142$

Subject to:

2) $V031 >= .9$

3) $V031 >= .69231$

4) $V031 >= .3077$

5) $V041 >= .9$

6) $V041 >= .1397$

7) $V041 >= .7982$

8) $V041 >= .0621$

9) $V042 - .0701V141 >= .8299$

10) $V042 - .0241V031 - .9618V041 >= -.0859$

11) $V042 >= .1944$

12) $V042 >= .7496$

13) $V042 >= .0691$

14) $V141 >= .9$

15) $V141 >= .1489$

16) $V141 >= .8511$

17) $V043 - .1V142 >= .8$

18) $V043 - .078V031 - .902V041 >= -.08$

19) $V043 - .994V042 >= -.094$

20) $V043 >= .2$

21) $V043 >= .72$

22) $V043 >= .098$

23) $V142 >= .9$

24) $V142 - .76V141 >= -.1$

25) $V142 >= .2$

26) $V142 >= .8$

References

[1] Heyman, D. P. and M. J. Sobel, *Stochastic Models in Operations Research, Vol. II* , McGraw-Hill Book Company, New York, NY, 1984.

[2] Knuth, D. E., "Optimal Binary Search Trees," *Acta Informatica*, 1 (1971),14–25.

[3] ————, "Binary Searching Trees," Chapter 6 in *Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

[4] Monahan, G. E., "Optimal Sequential File Search," *European Journal of Operational Research*, (1992, in press).

[5] Moore, J. C., W. B. Richmond, and A. B. Whinston, "A Decision Theoretic Approach to File Search," *Computer Science in Economics and Management* 1 (1988), 3–19.

[6] ———— and A. B. Whinston, "A Model of Decision-Making with Sequential Information-Acquisition (Part I)," *Decision Support Systems* , 2 (1986), 285–307.

[7] ———— and ————, "A Model of Decision-Making with Sequential Information-Acquisition (Part II)," *Decision Support Systems* , 3 (1987), 47–72.

[8] Wiederhold, G., "The Sequential File," Section 3-2 in *Database Design* , McGraw-Hill Book Company, New York, NY, 1977, 86- -94.